基于 P4 的 SDN 控制-数据平面流规则一致性校验 *

夏计强，崔鹏帅†，李子勇，兰巨龙

(中国人民解放军战略支援部队信息工程大学 信息技术研究所，郑州 450000)

摘　要：针对 SDN 数据平面的软硬件故障、错误配置等导致的控制平面和数据平面流规则不一致的问题，提出了基于 P4 的控制-数据平面流规则一致性校验机制(P4CV，P4-based Consistency Verification mechanism for SDN control-data plane)。P4CV 首先向数据平面发送特定结构的探针，然后各 P4 交换机将数据平面实际流规则执行信息嵌入到探针，最后 P4CV 采用基于符号执行的一致性校验算法，完成对控制平面流规则配置和数据平面遥测信息的一致性校验。仿真结果表明，P4CV 的单路径校验时长不受网络拓扑结构影响，仅与路径上交换节点数量线性相关。在同等网络规模和流规则配置的多路径转发场景中，P4CV 在仅产生约 0.06‰带宽开销的同时，比现有方案平均减少了约 42%的校验时长。

关键词：软件定义网络；P4；一致性；带内网络遥测

中图分类号：TP393　　doi: 10.19734/j.issn.1001-3695.2021.12.0694

P4-based rules consistency verification for SDN control-data plane

Xia Jiqiang, Cui Pengshuai†, Li Ziyong, Lan Julong

(*Information Technology Research Institute*, *People's Liberation Army Strategic Support Force Information Engineering University*, *Zhengzhou* 450000, *China*)

Abstract: To solve the inconsistency of flow rules between the control plane and data plane caused by SDN data plane software and hardware faults, configuration faults, etc. , this paper proposed a P4-based Consistency Verification mechanism for SDN control-data plane (P4CV) . P4CV firstly sent probes with a specific structure to the data plane, and then each P4 switch inserted its runtime information of flow rules in the data plane into the probes. Finally, P4CV adopted the consistency-verification algorithm, based on symbolic execution, to compare the configuration from the control plane and telemetry information from the data plane. The simulation results show that the single path verification time of P4CV is not affected by the network topology and is only linearly related to the number of nodes on the path. In a multipath forwarding scenario, with the same network scale and traffic rules configured, P4CV generates only 0.06‰ of bandwidth overhead and reduces the verification time by 42% compared with the existing scheme.

Key words: SDN; P4; consistency; In-band network telemetry

0　引言

传统网络架构中控制平面与数据平面紧耦合，软硬件的更新升级相互制约。软件定义网络[1](SDN, software defined network)通过将网络的管理控制逻辑从设备中抽离出来，解耦控制平面和数据平面，构建了一种高效管控的新型网络体系结构。SDN 在设计之初的可编程性主要体现在控制层。之后随着 P4[2]、POF[3]等数据平面领域特定语言(DSL，domain specific language)的出现，SDN 实现了线速处理性能的可编程数据平面。网络开发人员可以利用 P4 的抽象转发模型，灵活地设计数据平面的处理逻辑。基于此，数据平面对新协议的部署更加高效，在流量工程、网络测量等领域中得到了广泛应用[4,5]。但 P4 为数据平面带来可编程性的同时，也加剧了 SDN 控制平面和数据平面的一致性安全问题，即数据平面的实际转发行为与控制平面预期的行为不一致。一方面，作为一种数据平面编程语言，P4 本身的程序漏洞可能会导致控制-数据平面出现不一致的情况[6]；另一方面，当网络运行过程中出现软硬件故障、网络管理员错误配置或者遭到恶意攻击时，数据平面仍可能出现与控制平面预期行为不一致的安全问题。

在 P4 等数据平面编程语言出现之前，SDN 控制平面和数据平面流规则一致性问题已经得到了广泛关注[7~12]。Monocle[7]通过将交换机转发表逻辑表述为布尔可满足性问题来快速生成针对特定规则的探针，不仅能够完成对网络稳态下已加载的流规则的校验，还能够跟踪重配置时更新的流规则的加载情况。针对文献[8]中指出的商用交换机中存在的流规则优先级故障问题，Rulescope[9]进一步提出了更为精确高效的一致性校验算法。VeriDP[11]通过收集探针的路径信息并与控制平面的抽象路径表进行比对，有效地校验了真实网络流量的传输路径，但该方案需要预先构建路径表，增加了控制器的资源消耗。上述基于主动探测的一致性校验机制，都能够对控制器已知或已同步的流规则完成校验，但无法检测数据平面未知的或错误配置的流规则所导致的故障。同时，此类一致性校验机制是针对基于 OpenFlow 的 SDN 流规则校验场景设计的，而不适用于目前的可编程数据平面场景。

针对 SDN 可编程数据平面流规则一致性校验场景，近年来研究人员提出了一系列 P4 程序漏洞检查方案[13~15]。与传统的程序分析方法类似，这些方案主要通过对 P4 源代码的断言检查，实现对 P4 程序的静态漏洞分析。显然，此类方

收稿日期：2021-12-13；**修回日期：**2022-01-25　**基金项目：**国家重点研发计划(2019YFB1802501)；国家自然科学基金资助项目(61872382)

作者简介：夏计强(1996-)，男，安徽宿州人，硕士研究生，主要研究方向为新型网络体系结构、可编程数据平面；崔鹏帅(1990-)，男，河南安阳人，助理研究员，主要研究方向为新型网络体系结构、可编程数据平面(Mainblack@126.com)；李子勇(1995-)，男，安徽蚌埠人，博士研究生，主要研究方向为软件定义网络内生安全架构；兰巨龙(1962-)，男，河北人，教授，博导，主要研究方向为新一代信息网络关键理论与技术、信息网络安全.

法无法检测程序编译后运行时出现的错误和导致的不一致问题。因此，可编程数据平面领域的最新研究尝试围绕网络实际运行时的一致性问题开展工作。文献[16]开创了对 P4 支持的智能网卡的控制-数据平面不一致性的研究，该工作分析了只关注 P4 程序漏洞会对网络延迟、吞吐量等性能造成的严重影响。P4RL[17]通过引入基于强化学习的模糊测试，实现了单个 P4 交换机在-运行时的自动化验证，但该机制不适用于 SDN 网络规模的控制-数据平面的一致性校验。该团队进一步提出了一种面向 P4 SDN 的控制-数据平面一致性校验方法 P4Consist[18]。与 SDN 中数据平面测量的传统方法类似[7,10]，P4Consist 需要向网络注入大量的探针以检查所有规则或路径，不仅对网络正常通信造成影响，还会增加数据平面测量时延，进而使测试结果缺乏实时性。同时，由于 P4Consist 需要遍历源目节点间的所有路径，其校验时长随网络结构的复杂化急剧增长。

因此，如何快速完成对运行时的 SDN 可编程数据平面场景下的流规则一致性校验，成为可编程数据平面领域亟待解决的安全难题。对此，本文提出了基于 P4 的 SDN 控制-数据平面流规则一致性校验机制(P4CV, P4-based rules Consistency Verification mechanism for SDN control-data plane)，并基于 BMv2 软件交换机[19]完成了原型系统的开发和评估。主要贡献如下：(1)提出了基于 P4 的 SDN 控制-数据平面流规则一致性校验机制，完成了对运行时的 SDN 可编程数据平面转发行为的快速一致性校验；(2)提出了基于 P4 的网络遥测机制和基于符号执行的一致性校验算法，完成了对可编程数据平面流规则执行信息的快速收集和高效准确的一致性校验；(3)搭建了仿真模拟系统，实验结果显示，在单路径校验场景中，P4CV 校验时长不受网络拓扑结构复杂性影响，仅与校验路径上的交换节点数量线性相关。在多路径校验场景中，P4CV 在仅产生约 0.06‰带宽开销的情况下，比 P4Consist[18]平均减少了约 42%的校验时长。

1　相关背景和研究动机

1.1　P4 抽象转发模型

不同于 SDN 的传统 OpenFlow 模型[1]，P4 抽象转发模型设计了可编程的解析器和包处理动作，支持自定义的协议类型，实现了对数据报文的灵活处理。在[2]中给出的 P4 抽象转发模型中，交换机转发数据包的过程主要依赖可编程的数据包解析器、多级匹配-动作流水线和缓冲区三部分组件。

a) 解析器。在 P4 转发模型中，解析器首先对数据包进行处理，将包头从数据包中提取出来，与余下的载荷分开缓存。管理员可以定制数据包头结构和解析流程，解析流程会被编译器编译为数据包头解析图并配置到解析器上，被提取的包头部分按照解析图进行解析。

b) 多级流水线。经解析器解析后的数据包头部会经过多级匹配-动作表，这些匹配-动作表可以顺序、并行或者二者结合的方式执行，以流水线的形式组织起来，分为入口流水线和出口流水线。入口流水线用于确定数据包的输出端口和队列，出口流水线则负责修改数据包头部的信息。从逻辑上看，P4 的多级流水线是由一系列匹配-动作表(Match-Action Table)组成的一个有向无环图。加载 P4 程序的底层平台对数据包的处理过程严格按照有向无环图的逻辑进行。

c) 缓冲区。缓冲区用于临时存储已解析但尚未进入流水线处理阶段的数据包头部，以及解析后余下的载荷部分。

从 P4 程序的工作流程来看，P4 抽象转发模型包括配置阶段和运行阶段。配置阶段主要是设计解析器解析流程，设置匹配-动作表的执行顺序，并指定每个匹配-动作表处理的头部字段。这些配置决定了交换机支持哪些协议以及交换机如何处理数据包。运行阶段可以向配置阶段指定的匹配-动作表中增删条目，并在特定时间调用配置阶段生成的运行时控制接口(P4Runtime)，将匹配规则下发至数据平面，以便将配置策略作用于数据报文。

1.2　研究动机

在 SDN 网络运行过程中，网络内部和外界的种种扰动都可能会引起可编程数据平面的不一致转发行为(如硬件故障、恶意攻击等)。图 1 给出了一个典型的 SDN 可编程数据平面不一致转发行为的示例。控制平面通过 P4Runtime 接口为数据平面配置了一条转发路径 $S_1 \rightarrow S_3 \rightarrow S_2 \rightarrow S_4$ (图中实线所示)，以及相应的 P4 处理逻辑和流规则。网络管理人员在交换机 S_3 中配置了防火墙规则，以保护下游的服务器资源。如果交换机 S_1 受到恶意攻击(或未知的硬件故障等)使流规则被窜改，致使流量的实际转发路径转为 $S_1 \rightarrow S_2 \rightarrow S_4$ (图中虚线所示)。这会导致流量绕过交换机 S_3 中的防火墙，进而可能对下游的服务器资源造成损害。

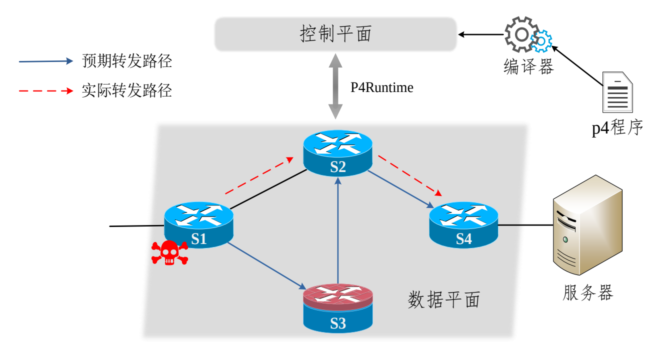


图 1　可编程数据平面不一致转发示例

Fig. 1　An example for the inconsistent forwarding behavior of the programmable data plane

此外，流规则优先级错误[8,20]同样会导致类似的不一致转发行为。例如：PicOS 2.1.3 中设定了交换机在硬件表满时用软件缓存流表。对于一组转发路径重叠的流规则 r_1 (高优先级)和 r_2 (低优先级)，如果 r_1 缓存在交换软件中而 r_2 存储在硬件表中，则流量会违背流规则的优先级，执行 r_2 的匹配-动作表中的指定动作。类似地，HP 5406zl 商用交换芯片不支持流规则优先级，对数据报文的处理始终按照最新的流规则而忽略优先级。这些情况会导致网络管理人员开发的可编程数据平面处理逻辑无法得到有效和一致地执行。

2　P4CV 设计

2.1　总体框架

由于上述可编程数据平面的不一致转发行为是在网络运行时产生，因此控制平面无法通过针对 P4 程序的静态分析方法来获取这些危及网络安全的错误配置。对此，本文提出了面向可编程数据平面的一致性校验方法 P4CV。

P4CV 工作流程如图 2 所示。首先，由流量生成器向网络数据平面输入探针流，为降低对网络带宽的影响，探针由简单的五元组信息(即源/目 IP 地址、源/目 MAC 地址及传输层协议)和必要的源路由端口序列构成。然后，探针流依据数据平面流规则配置和指定的源路由端口序列到达目的节点，途中经过的每一个交换机都将五元组匹配的流规则信息添加到探针中。之后，当探针流到达目的节点时，P4CV 采用现有的 sFlow[21]方法对探针流进行采样。探针收集的遥测数据(包括实际的转发路径和流规则信息)被解析出来并交付给一致性校验模块。最后，一致性校验模块依据数据平面收集的遥测数据和控制平面返回的可达路径图，通过采用对转发路径分析和符号执行的方式，完成对数据平面转发行为一致性的校验。

P4CV 由以下四个功能模块构成：

a) 数据平面模块。设计实现基于 P4 的数据包处理逻辑，完成对网络流量的转发，并将遥测数据压入探针；

b) 控制平面模块。对给定的探针和源目节点，依据网络拓扑和配置信息，返回可达路径图；

c) 一致性校验模块。依据可达路径图和遥测数据，完成对数据平面转发行为一致性的校验；

d) 输入/输出模块。 P4CV 的输入为一个探针流量生成器，输出为转发路径上所有的不一致信息(或数据平面不存在不一致转发行为)。

下面对各个功能模块进行详细介绍。

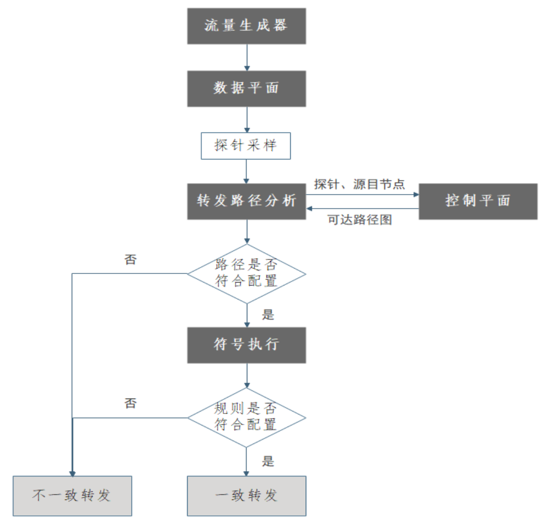


图 2 P4CV 工作流程

Fig. 2 P4CV workflow

2.2 数据平面模块

为减少遥测过程对网络正常通信造成的影响，同时提高流规则一致性校验的准确性，本文提出了一种高效的基于 P4 的网络遥测机制。传统的带内网络遥测(INT)[22]是一种被动遥测技术，采用将遥测指令和数据封装到正常数据分组中的方式,探测路径上特定的遥测数据。这样的方式降低了数据分组的有效载荷比，增加了数据平面开销，对网络中的正常数据通信造成影响。对此，本文通过采用特定数量和结构的探针，来收集正常数据分组转发过程匹配的流规则 ID、出入端口号等流规则执行信息，在简化探针结构、降低数据平面开销的同时，实现了对一致性校验所需的必要信息的收集。

在基于 P4 的可编程数据平面中，数据包的解析过程被抽象为一个有限状态机(FSM, finite state machine)，每个状态节点代表数据包头的一个字段，边代表状态间的切换。解析器按序遍历数据包头，并在执行过程中提取各字段值，由预定义的匹配-动作表进行处理。P4CV 对探针的解析过程由图 3 中的解析图(parser graph)表示。从图中可以看出，P4CV 数据平面中各交换节点对网络中数据分组的处理包含两种逻辑:

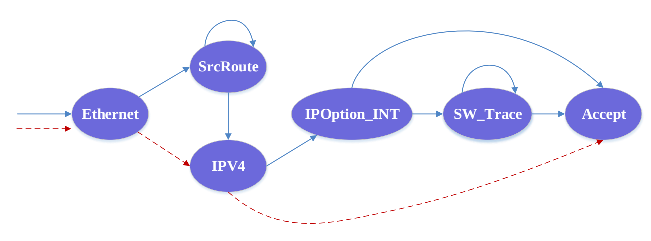


图 3 报文处理程序解析图

Fig. 3 Parse graph of packet processing program

a) 对正常数据分组的处理。与传统网络中对 IP 数据包的处理流程一致(图 3 中虚线)，P4CV 依据三层路由或二层交换转发正常的数据分组，以保证网络正常的通信业务。

b) 对探针的处理。如图 3 中实线所示，P4CV 依据探针包头中的源路由端口序列(*SrcRoute* 字段)，从指定的出端口转发探针。同时在转发探针之前，将遥测数据嵌入探针包中。这些数据被记录在 *SW_Trace* 字段，包括当前交换机 ID、探针包入端口号，和探针依据三层路由或二层交换进行转发时匹配的流规则 ID 以及相应的出端口号，即真实数据流转发时流规则执行情况，具体格式将在 2.5 节中详细阐述。

2.3 控制平面模块

在 P4CV 中，控制平面主要提供一致性校验的对照数据(包括转发路径和流规则)，它将网络配置抽象为一个三元组结构 $G=\{V,E,R\}$。其中，V 表示网络中的所有节点，E 表示 V 中节点间的有向边，R 表示转发规则。对于 G 中的一条有向边(s, t)，$R(s, t)$表示节点 s 中存在 $s\rightarrow t$ 的转发规则。对于给定的探针和源目节点，控制平面模块依据网络拓扑和流规则配置信息，将三元组结构的源目交换节点间的可达路径图交付给一致性校验模块以完成最终的校验工作。

此外，P4 中的控制平面下发给数据平面的流规则存储在相应的配置文件中(通常为 JSON 格式)，各交换节点对应于彼此间相互独立的配置文件。这些文件包含各交换节点对应的所有匹配-动作表(即转发规则)的名称、参数等信息。控制平面模块负责解析其存储的交换机配置文件信息。当一致性校验模块从数据平面接收到探针(五元组流)时，由控制平面解析配置文件信息，解析结果转存为一个字典，其中键是交换机 ID，值是该交换机的 JSON 配置文件中所有可用规则的一个列表。最后由一致性校验模块以符号执行的方式，对控制平面解析后的配置信息和数据平面遥测数据进行比对，以检测控制-数据平面的不一致性。

本文通过对控制平面和数据平面采用模块化设计，延续了 SDN 网络中将控制平面与数据平面解耦的基本准则。因此，控制平面如何管理数据平面各网络设备相关规则的方式是设备无关的。实际上，基于 JSON 格式的配置文件是特定于本文实验采用的软件交换机模型(BMv2)[19]，而本文所提一致性校验方法是通用的。如果控制平面使用其他文件格式来存储转发规则，只需要调整控制平面模块中的解析方式即可。

2.4 一致性校验模块

一致性校验模块负责比对来自数据平面的遥测数据和控制平面的配置信息，以完成对数据平面转发行为一致性的校验。一致性校验模块由转发路径分析和符号执行两部分功能组成。

如图 2 所示，当收到数据平面遥测数据时，一致性校验模块向控制平面模块发送探针信息并获取对应的可达路径图。发送的探针信息包括探针包结构和给定的源目节点，其格式为*[packet, src, dst]*。转发路径分析功能模块采用 DFS 搜索算法，以 *src* 为起点遍历可达路径图，获取*(src, dst)*间所有符合网络配置的路径，进而判断探针遥测路径是否为给定源目节点间符合配置的转发路径。同时，为加快遍历过程，搜索算法采用剪枝 DFS 算法(depth 为探针转发路径的长度)。

为提高 P4CV 校验结果的准确性，一致性校验模块在完成对转发路径的校验后，还需要对转发过程匹配的流规则进行校验。符号执行功能模块根据探针包五元组信息生成一个具有相同报头的符号包(SP, symbolic packet)，然后沿探针转发路径逐跳模拟该 *SP* 的转发过程。对 *SP* 的模拟转发过程采用上述布尔函数实现，已在 2.2 节中详细阐述。对转发路径图中的一条有向边(u, v)，如果节点 u 中存在转发规则 $r(u, v)$ 且与控制平面流规则一致，则将节点 u 标记为 *True* 并更新 *SP*。校验算法如下:

算法 1 一致性校验算法(符号执行)

输入:符号包 SP，校验路径 path_spec，流规则配置信息 Rules。

输出：校验结果 PATHS_CHECK,不一致的交换节点信息 Error_Report。

```
for switch∈path_spec do
  if (last switch) then
    for rule∈Rules do
      if check (SP, rule) == TRUE then
        PATHS_CHECK ← TRUE //该路径不存在一致性问题
      elseif (last rule) then
         SWITCH_CHECK ← False
         PATHS_CHECK ← False //不一致路径
         Error_Report
  else
     for rule∈Rules do
       if check (SP, rule) == TRUE then
          SWITCH_CHECK ← TRUE
          Go to next switch //继续校验下一节点
       elseif (last rule) then
          SWITCH_CHECK ← False
          PATHS_CHECK ← False //不一致路径
          Error_Report
          Go to next switch //继续校验下一节点
```

此外，为了模拟数据包在交换机中实际的查表转发行为，符号执行采用顺序查表的方式(不考虑流表优先级的情况)。同时，为保证校验结果的准确性，P4CV 对流规则信息的校验包括目的 IP、出/入端口号、流规则 ID 等。如果流规则匹配，则该交换机被标记为 *True*，并更新 *SP* 中各字段信息，之后继续校验下一节点；如果流规则匹配失败，则该节点和相应转发路径被标记为 *False*，输出故障信息之后，同样继续校验下一节点，以检测该路径上其他交换节点是否存在流规则不一致错误。

2.5　输入/输出模块

本节将介绍 P4CV 的输入/输出模块。

1) 输入模块

P4CV 采用 INT[22]的方式来获取数据平面实际的转发行为信息，需要向网络中输入探针流。因此，本方案的输入模块即为探针流量生成器。为了避免探针流占用过多的链路带宽，每个探针仅包含基本的五元组信息和必要的源路由端口序列，探针包格式如图 4 所示。其中，探针包的转发路径由给定的源路由端口序列确定(即指定探针在各交换节点的出端口)，五元组信息则用于确定真实数据流依据三层路由或二层交换转发时所匹配的流规则信息，各信息字段设置如下：

a) SrcRoute(*n* bytes)。源路由端口序列，以栈结构存储探针在各交换节点转发时的出端口(7bits)，以及相应的栈底标志位 *bos*(1bit)，占 *n* bytes($n \le N$,*N* 为路径上交换节点数量)。

b) IPOption_INT(4 bytes)。IP 可选字段的遥测数据标志字段，同时记录简要的遥测信息。

c) SW_Trace(4 bytes)。记录转发过程中收集的流规则信息,包括节点 ID、流规则 ID 和出入端口号等。

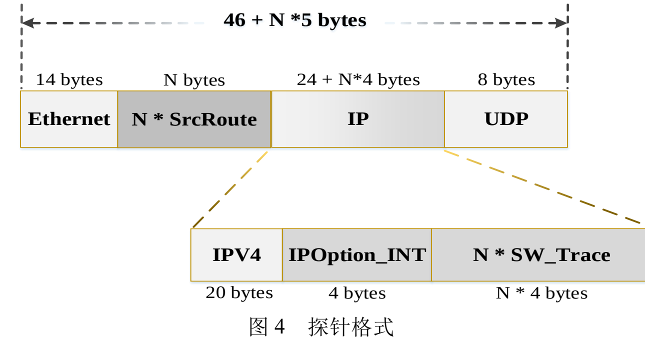


图 4　探针格式

Fig. 4　Probe format

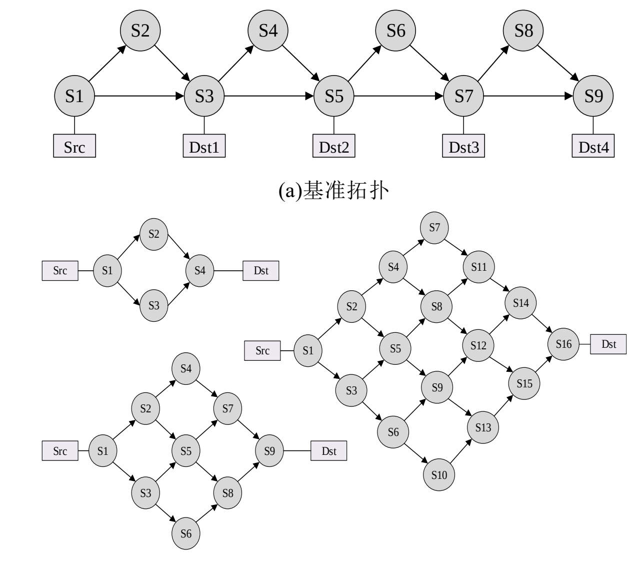
2) 输出模块

本方案在一致性校验过程中，会逐个比对探针经过的每一台交换机，并标记出现流规则不一致的交换机。在一致性校验模块完成一致性校验工作后，将输出每条转发路径的校验结果。若其中某些路径存在不一致的情况，则同时输出该路径相应故障节点信息(包含节点 ID、流规则信息等)。此外，本文的一致性校验算法在发现路径上不一致交换节点时，仍会继续校验路径下游其他交换节点。因此在给定路径存在不一致情况时，会输出该路径所有的故障节点信息。

3　实验评估

3.1　实验设置

为验证本文提出的可编程数据平面一致性校验机制的可行性和有效性，本文设置了两组实验：图 5 所示的单路径校验场景和图 6 所示的多路径校验场景。其中，第一组实验以图 5a 所示拓扑的实验结果作为单路径校验场景的基准值，第二组实验采用目前数据中心广泛部署的胖树拓扑(fat-tree)。每组实验以 10 次实验的结果的平均值作为最终结果。



(a)基准拓扑

(b)不同节点数量的栅格拓扑(4/9/16)

图 5　单路径场景实验拓扑

Fig. 5　Topologies of the single-path experiment

实验环境和参数设置如下：实验选用 BMv2 软件交换机[19]构建网络拓扑，在虚拟机中的 Mininet[23]仿真器上进行测试。虚拟机配置为 Ubuntu 18.04-LTS 操作系统，搭载 Intel Core i7-9700 3.00GHz 处理器，8G 内存。在单次实验中，分别给各网络交换节点配置了相同数量的转发规则(15K,30K,60K)。为保证校验结果的实时性，同时减小探针流发送时延，源节点探针发送速率设定为 100pps。此外，实验中错误配置的注入采用与图 1 中示例相同的方式，即通过可编程交换机的命令行配置接口修改其中的流规则。这些错误配置随机地分布在转发路径上的各节点中。

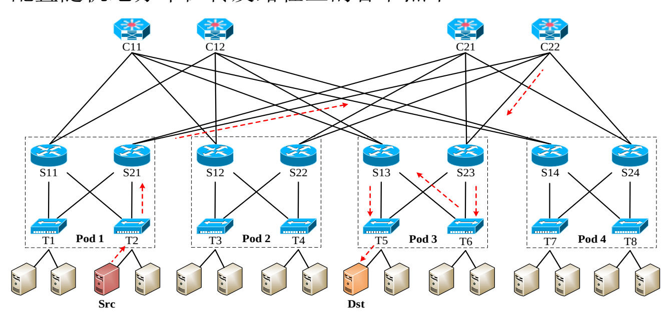


图 6　多路径场景实验拓扑

Fig. 6　Topology of the multi-path experiment

3.2　单路径校验

为了评估本方案在单路径场景下的一致性校验性能，搭建了图 5 所示网络拓扑。图 5a 中，S1 作为源节点，S3、S5、S7、S9 依次作为目的节点。首先由源节点将给定单条路径对应的源路由端口序列压入探针头部(格式如图 4)，然后周期性发送一组探针，目的节点收到探针后进行采样、解析并完成一致性校验。从探针发出到 P4CV 完成一致性校验的时间记为完成单条路径校验的总时长 *T*，校验过程中的符号执行时长记为 *T_se*。从图 7 中实验结果可以看出，单条路径符号执行时间和校验总时长都与该路径上交换节点的数量线性相关。由于符号执行采用顺序查表的方式，且每台交换机内配置了相同数量的流规则，因此符号执行时长以及配置文件解析时延均与路径上交换机数量线性相关。同时，由于探针遥测时间为微秒级(以 1Gbps 链路计算)，可忽略不计，因此校验总时长同样与路径交换机数量线性相关。从图 7a 中可以看出，当校验路径上的节点不超过 8 个、单个节点流规则数量为 15K 时，P4CV 能够在 30s 内完成对所有错误配置的精确定位，校验结果具有较强的实时性和可靠性，从而保证了校验结果的准确性。

此外，为进一步验证 P4CV 中控制-数据平面一致性校验时长仅与路径上交换节点数量线性相关，而与网络拓扑结构无关，本文同样在图 5b 的网络拓扑中进行了单路径一致性校验。从表 1 的实验结果可以看出，在转发路径上交换节点数量相同的情况下，P4CV 在不同网络拓扑中的单路径校验时长基本相等，误差范围不超过 3%。

表 1　各拓扑单路径校验总时长

Tab. 1　Total verification time of single-path scenario of each topology

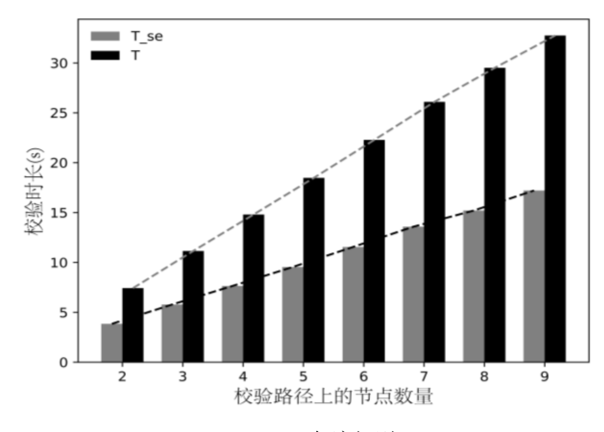
流规则数	校验总时长/s					
	baseline	grid_4sw	baseline	grid_9sw	baseline	grid_16sw
15K	11	11	15	15	27	27
30K	24	24	39	40	55	57
60K	46	47	78	80	108	111

3.3　多路径校验

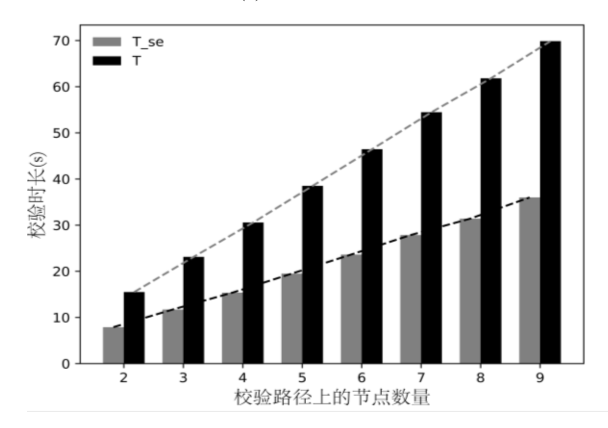
数据中心网络是可编程数据平面目前应用最广泛的网络场景。因此，本文实验搭建了图 6 所示的数据中心网络的胖树拓扑(*pod=4*)，以验证本文提出的可编程数据平面一致性校验机制在实际部署中的可行性和有效性。该拓扑由核心层、汇聚层和边缘层的各交换节点构成，包括 4 个核心节点、8 个汇聚节点和 8 个边缘节点，每个边缘节点连接 2 台服务器。在经过剪枝 DFS 算法计算后(*depth*=7)，该拓扑中任意一对不同 Pod 下的边缘交换节点间存在 20 条不同的转发路径。

如图 6 所示，实验选取 pod1 和 pod3 中的两台服务器分别作为探针的发送和接收节点。图中给出了两节点间的一条转发路径(图中虚线所示)。每次实验通过 bash 脚本在源目节点间的 *n* 条转发路径中注入 20 个错误配置($2 \le n \le 20$)，并记录检测到每个错误配置所需要的时间。

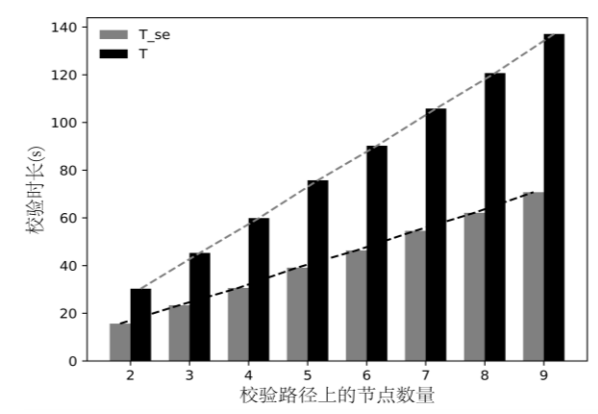
图 8 给出了实验中 P4CV 和 P4Consist 两种校验方法的错误配置检测时间的累积分布函数(CDF，Cumulative Distribution Function)。从图中可以看出，二者均能够完成对网络中所有错误配置的检测，但 P4CV 始终比 P4Consist 先检测到各路径上的错误配置。通过引入源路由转发机制，P4CV 加快了探针流对转发路径的遍历过程。同时，P4CV 的一致性校验算法在发现路径中存在错误配置时，仍继续校验转发路径上的其他下游节点。因此，单个探针足以完成对单条路径的一致性校验工作，极大地缩短了探针流对错误配置的探测时间。此外，从图 8 的实验结果可以看出，二者校验总时延的差值随节点中流规则数量的增多而扩大。在各交换节点中流规则数量分别为 15K、30K、60K 时，P4Consist 分别需要 7mins、18mins 和 36mins 以完成对所有错误配置(20 个)的检测。而对于同等规模的网络配置，P4CV 分别只需要 4mins、10mins 和 20mins 的检测时间，检测效率平均提高了 42%。



(a) 15K 条流规则



(b) 30K 条流规则



(c) 60K 条流规则

图 7　单路径实验结果

Figure 7 Results of the single-path experiment

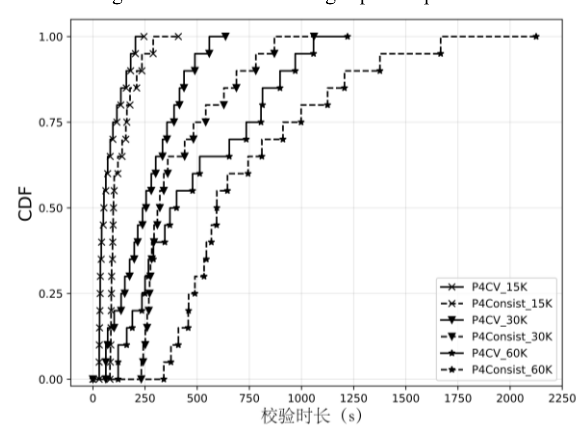


图 8　多路径实验结果

Fig. 8　Result of the multi-path experiment

3.4 数据平面开销

不同于传统方案依靠探针丢包率、延迟等指标的校验方式，本方案数据平面采用基 P4 的网络遥测机制收集数据平面的实际转发行为信息。由于源路由机制明确了遥测路径，单个探针足以收集单条路径的流规则信息。探针的格式如图 3 所示，单个探针总长度最长为 $46 + 5N$ *bytes*(N 为路径上交换机个数)。以图 6 中的 4 层胖树拓扑为例($N = 7$)，探针长度最长为 84 bytes，探针流以 100pps 的速率注入网络，对 1Gbps 链路带宽最高仅产生约 0.06‰的带宽消耗。

4 结束语

本文针对 SDN 中控制平面和数据平面流规则不一致问题，提出了基于 P4 的 SDN 控制-数据平面流规则一致性校验机制 P4CV。P4CV 采用了基于 P4 的网络遥测机制，充分发挥了可编程数据平面包处理过程灵活可定义的优势，进而完成了对数据平面流规则执行信息的快速收集。同时，本文提出了基于符号执行的一致性校验算法，完成了对给定源目交换节点间单路径和多路径转发场景的流规则一致性校验工作。仿真结果表明，与现有的流规则一致性校验机制相比，P4CV 的单路径校验时长仅与路径上交换节点数量线性相关，而不受网络拓扑结构变化影响。同时对于多路径转发场景的流规则一致性校验，P4CV 在进一步降低数据平面开销的同时，大幅提高了校验效率。此外，本文所提方案面向基于 P4 的 SDN 网络场景设计，其核心校验模型能够在所有支持 P4 编程的交换设备中实现快速部署，具有良好的可扩展性。后续工作将继续优化校验算法，进一步降低校验时长。

参考文献：

[1] McKeown N, Anderson T, Balakrishnan H, *et al.* OpenFlow: enabling innovation in campus networks [J]. ACM SIGCOMM Computer Communication Review, 2008, 38 (2): 69-74.

[2] Bosshart P, Daly D, Izzard M, *et al.* P4: programming protocol-independent packet processors [J]. ACM SIGCOMM Computer Communication Review, 2013, 44 (3): 87-95.

[3] Song Haoyu. Protocol-oblivious forwarding: Unleash the power of SDN through a future-proof forwarding plane [C]// Proc of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking. New York: ACM Press, 2013: 127-132.

[4] Hauser F, Häberle M, Merling D, *et al.* A survey on data plane programming with P4: fundamentals, advances, and applied research [J]. arXiv Preprint arXiv: 2101. 10632, 2021.

[5] 林耘森箫, 毕军, 周禹, 等. 基于 P4 的可编程数据平面研究及其应用 [J]. 计算机学报, 2019, 42 (11): 22. (Lin Yunsenxiao, Bi Jun, Zhou Yu, *et al.* Researches and applications of programmable data plane based on P4 [J]. Chinese Journal of Computers, 2019, 42 (11): 22.)

[6] Dumitru M V, Dumitrescu D, Raiciu C. Can we exploit buggy P4 programs? [C]// Proc of the Symposium on SDN Research. New York: ACM Press, 2020: 62-68.

[7] Perešíni P, Kuźniar M, Kostić D. Monocle: dynamic, fine-grained data plane monitoring [C]// Proc of the 11th ACM Conference on Emerging Networking Experiments and Technologies. New York: ACM Press, 2015: 1-13.

[8] Kuźniar M, Perešíni P, Kostić D. What you need to know about SDN flow tables [C]// International Conference on Passive and Active Network Measurement. Berlin: Springer, 2015: 347-359.

[9] Wen Xitao; Bu Kai, Yang Bo, *et al.* Rulescope: inspecting forwarding faults for software-defined networking [J]. IEEE/ACM Trans on Networking, 2017, 25 (4): 2347-2360.

[10] Zhao Yu, Wang Huazhe, Lin Xin, *et al.* Pronto: efficient test packet generation for dynamic network data planes [C]// IEEE the 37th International Conference on Distributed Computing Systems. Brussels: IEEE Computer Society, 2017: 13-22.

[11] Zhang Peng, Li Hao, Hu Chengchen, *et al.* Mind the gap: monitoring the control-data plane consistency in software defined networks [C]// Proc of the 12th International on Conference on emerging Networking Experiments and Technologies. New York: Association for Computing Machinery, 2016: 19-33.

[12] 赵会, 吕光宏, 杨洋, 等. SDN 故障分析研究综述 [J/OL]. 计算机应用研究, 2020, 37 (10) . (2019-10-24) [2021-10-19]. https://www. arocmag. com/article/01-2020-10-003. html (Zhao Hui, Lyu Guanghong, Yang Yang, *et al.* SDN fault analysis research. [J/OL]. Application Research of Computers, 2020, 37 (10) . (2019-10-24) [2021-10-19]. https://www. arocmag. com/article/01-2020-10-003. html)

[13] Liu J, Hallahan W, Schlesinger C, *et al.* P4v: practical verification for programmable data planes [C]// Proc of the 2018 Conference of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2018: 490-503.

[14] Freire L, Neves M, Leal L, *et al.* Uncovering bugs in p4 programs with assertion-based verification [C]// Proc of the Symposium on SDN Research. New York: ACM Press, 2018: 1-7.

[15] Stoenescu R, Dumitrescu D, Popovici M, *et al.* Debugging P4 programs with Vera [C]// Proc of the 2018 Conference of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2018: 518-532.

[16] Gray N, Grigorjew A, Hosssfeld T, *et al.* Highlighting the gap between expected and actual behavior in p4-enabled networks [C]// IFIP/IEEE Symposium on Integrated Network and Service Management. Piscataway, NJ: IEEE Press, 2019: 731-732.

[17] Shukla A, Hudemann K N, Hecker A, *et al.* Runtime verification of p4 switches with reinforcement learning [C]// Proc of the 2019 Workshop on Network Meets AI & ML. New York: ACM Press, 2019: 1-7.

[18] Shukla A, Fathalli S, Zinner T, *et al.* P4Consist: toward consistent P4 SDNs [J]. IEEE Journal on Selected Areas in Communications, 2020, 38 (7): 1293-1307.

[19] Behavioral Model Repository. P4 Language Consortium [EB/OL]. [2021-10-11]. https://github. com/p4lang/behavioral-model.

[20] Zhang Peng, Zhang Cheng, Hu Chengchen. Fast data plane testing for software-defined networks with RuleChecker [J]. IEEE/ACM Trans on Networking, 2018, 27 (1): 173-186.

[21] Phaal P, Panchen S, McKee N. InMon corporation's sFlow: a method for monitoring traffic in switched and routed networks [S]. RFC3176, 2001.

[22] Zhu Yibo, Kang Nanxi, Cao Jiaxin, *et al.* Packet-level telemetry in large datacenter networks [C]// Proc of the 2015 ACM Conference on Special Interest Group on Data Communication. New York: ACM Press, 2015: 479-491.

[23] Pal C, Veena S, Rustagi R P, *et al.* Implementation of simplified custom topology framework in Mininet [C]// Asia-Pacific Conference on Computer Aided System Engineering. Piscataway, NJ: IEEE Press, 2014: 48-53.